Multi-Step Strategies for Rendezvous Search on the Edges of the Platonic Solids

by

Xiao Xie

A thesis submitted to The Johns Hopkins University in conformity with the requirements for the degree of Master of Science in Engineering.

Baltimore, Maryland

May, 2019

Abstract

The *astronaut problem* is an open problem in the field of rendezvous search. The premise is that two astronauts randomly land on a planet and want to find one another. Research explores what strategies accomplish this in the least expected time. To investigate this problem, we create a discrete model which takes place on the edges of the Platonic solids. Some baseline assumptions of the model are: (1) The agents can see all of the faces around them. (2) The agents travel along the edges from vertex to vertex and cannot jump. (3) The agents move at a rate of one edge length per unit time. We first explore an unbiased random walk strategy where the agents move in a random direction on each turn. We then explore multi-step strategies, which are strategies where both agents move randomly for one step, and then follow a pre-determined sequence. We compare the performance of multi-step strategies and the unbiased strategy for all of the solids. For the cube and octahedron, we are able to prove optimality of the "Left Strategy", in which the agents move in a random direction for the first step and then turn left. For a dodechedron, we prove op-

ABSTRACT

timality of a multi-step strategy using a lower bound for the expected meeting time. For the icosahedron, we present results for a subset of the multi-step strategies. In an effort to find lower expected times, we explore mixed strategies. Mixed strategies incorporate an asymmetric case which under certain conditions can result in lower expected times. Due to the greater complexity of calculating the expected time of mixed strategy, we again utilize lower bounds to find bounds for the optimal expected time. Most of the calculations were done using first-step decompositions for Markov chains.

Primary Reader and Advisor: John Wierman

Acknowledgments

I would like to thank my research adviser, Professor John C. Wierman for his support over the past two and half years. Not only did he introduce me to mathematical research, but he provided countless opportunities to learn and grow as a researcher. I would also like to thank my collaborator, Elanor West. She has come up with a lot of interesting ideas for this work, and is a great presentation partner. I would also like to thank Jeffrey Braun for introducing me to the problem in the Introduction to Research in Discrete Probability course. I also thank the Acheson J. Duncan Fund for the Advancement of Research in Statistics for supporting this research and giving us the opportunity to share it at conferences. Lastly, I would like to thank the Applied Mathematics and Statistics department for all the inspiration, encouragement, and support I've received over the past four years. I can't imagine being part of a different department, and I am grateful for all the people I've met.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

The general premise behind all search theory problems is that an agent is looking for a target (which could be mobile or immobile). Thus, the question we are naturally led to ask is "What is the optimal strategy?" Optimal is traditionally defined to mean the strategy that minimizes expected meeting time. From this basic premise, a wide variety of problems have been proposed. Search theory originally found its roots in military operations during World War II. During the period from 1965 to 1975, Lawrence Stone pioneered mathematical results for searches involving a single agent and an immobile target and made important progress on problems involving a moving target [1]. Another class of problems which has been thoroughly researched are search games which in-

volve a searcher, who is looking to minimize time, and a hider, who is looking to maximize time [2].

Moreover, Thomas Schelling presents an interesting problem of ”tacit coordination” in his book, *The Strategy of Conflict*. Two parachutists randomly land in area with defining landmarks such as roads, buildings, a river, and a bridge. The goal for the parachutists is to find one another. Schelling makes the claim that the crux of the problem is the two parachutists must meet at a unique ”focal point”. He then goes on to say that the problem cannot be properly defined (and thus cannot be solved) when the search area is homogeneous, i.e., there are no distinguishing focal points [3]. We will see that rendezvous search, a subset of search theory, addresses this exact scenario.

Rendezvous search is a branch of search theory that has been gaining attention in recent years. The premise of rendezvous search problems is that two (or more) agents are in different locations, and they want to find one another. In this case, both agents want to minimize the time it takes to achieve this goal. Rendezvous search can be further broken down into two categories: asymmetric and symmetric. In the asymmetric case, the agents are not bound to the same strategy; in the symmetric case, they are. Steve Alpern, a pioneer in the field of rendezvous search, proposed ten open rendezvous search problems in *Search Theory: A Game Theoretic Perspective* [4, Ch. 14]. Our research is inspired by the astronaut problem.

The astronaut problem is as follows: two astronauts randomly land on a spherical planet, each with the same detection range in which they can see each other, and their goal is to find one another. No significant progress has been made on this problem. However, there are some results for rendezvous search on two-dimensional objects, such as graphs and networks, which could potentially be used to approximate the sphere. In this background, we present results for the asymmetric case and the symmetric case.

Significant work has been done for n discrete locations on a connected graph by Anderson and Weber. Results for the asymmetric case have found that a "Wait for Mommy" strategy is optimal [5]. This strategy entails that one agent waits while the other searches all the locations. For the symmetric case, one notable result is the Anderson-Weber Strategy for rendezvous search on n discrete locations in a complete graph. In this strategy, the agents have a probability of staying in the same location for $n-1$ steps and a probability of visiting all other $n-1$ locations for $n-1$ steps. If one agent waits and the other moves, then the agents are guaranteed to find one another. Meanwhile, if both agents move, there is no such guarantee. If both players wait, then they will definitely not meet. This strategy drew inspiration from the optimal "Waiting for Mommy" strategy in the asymmetric case, and has been proven optimal for two locations and three locations [5, 6].

Another variation of the problem that has been considered is the rendezvous

search problem on the circle. Two agents are on an undirected circle and want to find one another. One symmetric strategy that has been studied by Alpern is the *Coin Half Tour*. In this strategy, each agent flips a fair coin. If the coin is heads, then the agent will walk halfway around the circle in one direction. If the coin is tails, then the agents will walk halfway around the circle in the other direction. The agents repeat this process until they meet. On a circle with a circumference of 1, this strategy yields an expected time of $\frac{3}{4}$. Alpern also looked at the asymmetric "Wait for Mommy" strategy on the circle. Under the same conditions as before, this strategy yields an expected time of $\frac{1}{2}$ [7]. The "Coin Half Tour" and "Wait for Mommy" strategy are thought to be the best available strategy for the symmetric and asymmetric case respectively.

Along with the work presented in the mathematics community, the computer science community has also published results on variations of this problem. In the field of computer science, this problem considers two mobile agents who are located on two vertices of a network. The agents' goal is to meet at the same vertex. A plethora of variants have been considered. For example, in one version of asymmetric rendezvous, a map of the graph and the agent's starting position is available for reference. Algorithms, called FOCAL strategies, have been created to get the agents to an agreed upon meeting location. In another version, the agents do not have access to a map, but there is a distinguishing focal point in the graph. In one version of symmetric rendezvous, agents are

allowed to leave tokens to mark nodes as desired [8]. In addressing efficiency of strategies, researchers have considered both the expected time it takes for the agents to meet and the worst-case scenarios. Overall, the computer science work seems to emphasize utilizing all known information about the network in order to determine 1) Is the problem solvable? 2) What is the best solution?

1.2 The Discrete Model

In investigating the astronaut problem, we decided to simplify and approximate this problem with the five platonic solids: the tetrahedron, the octahedron, the cube, the icosahedron, the dodecahedron. Recall that the platonic solids are polyhedrons composed of congruent, regular faces with the same number of faces meeting at each vertex. The five solids listed above are the only polyhedrons that satisfy this property. Refer to Figures 1.1, 1.2, 1.3, 1.4, and 1.5 for images. These solids seem to be a unique model among the rendezvous search literature, which currently focuses on two-dimensional graphs as opposed to three-dimensional objects.

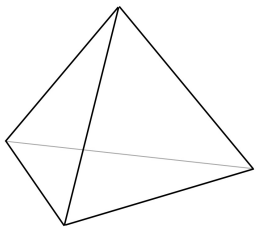

Figure 1.1: Tetrahedron.

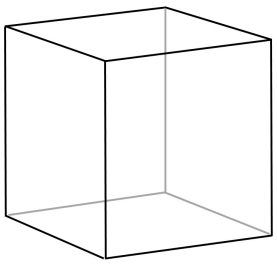

Figure 1.2: Cube.

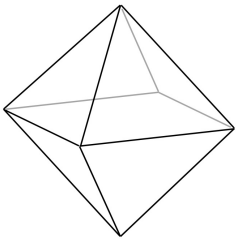

Figure 1.3: Octahedron.

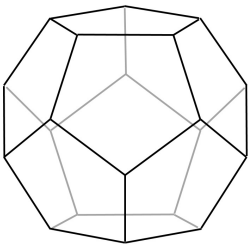

Figure 1.4: Dodecahedron.

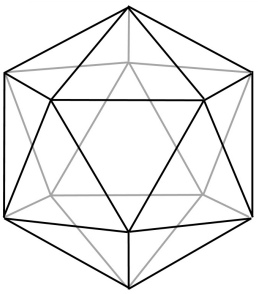

Figure 1.5: Icosahedron.

Each platonic solid has a unique set of properties. In Table 1.1, we list the face shape, number of vertices, number of edges, number of faces, and degree of vertices (number of edges incident to a vertex) for each solid.

Solid	Face Shape	Nodes	Edges	Faces	Node Degree
Tetrahedron	Triangular	4	6	4	3
Cube	Square	8	12	6	3
Octahedron	Triangular	6	12	8	4
Dodecahedron	Pentagonal	20	30	12	3
Icosahedron	Triangular	12	30	20	5

Table 1.1: Properties of the platonic solids.

One property that they share in common is they all are vertex-transitive, and thus from any vertex, the surrounding area looks the same. Therefore when analyzing movements, we only need to consider the distance between the agents. We hope to find strategies that can be adapted to the original astronaut problem on a sphere. Our intuition is that "larger" solids will more closely approximate the sphere, though it is still necessary to compare both "larger" and "smaller" solids.

We consider a rendezvous search problem with the following constraints:

1. The agents start on vertices and move along the edges from vertex to vertex. The agents cannot travel a fraction of an edge.

2. The agents both move at a rate of one edge length per unit of time. We also consider one edge length to be one unit of length.

3. The agents each have the same given detection range for which they can see each other.

4. The agents start in positions where they cannot see each other.

5. The agents cannot jump from one vertex to a non-adjacent vertex.

6. The search ends once the agents see each other.

Throughout this thesis, we assume that the agents have full-face visibility. Full-face visibility means agents can see all faces (and their vertices) adjacent to their current vertex. For example, an agent on the icosahedron can see five triangular faces, along with their edges and vertices, and an agent on the cube can see three square faces, along with their edges and vertices. On each of the five solids, an agent can see at least half of the nodes of the solid from any given node.

Our ultimate goal is to minimize the expected time for the agents to see each other.

1.3 Outline

In chapter 2, we introduce and discuss unbiased random walks on the edges of the five solids. These strategies act as an initial baseline for comparison. We calculate these expected times using first-step Markov chain decompositions.

However, you can also use geometric probability mass functions. Intuitively, an unbiased random walk strategy can be considered "mindless" and thus, we would expect it to give longer expected meeting times than more intentional strategies. However, we see later that this is not always the case.

In Chapter 3, we discuss multi-step strategies on the solids. For these strategies, agents move in a random direction for the first step, and then follow a predetermined sequence. Our results lead us to discuss what it means to be "optimal," since a low expected meeting time does not necessarily guarantee that the search will not take an arbitrarily long time. On the other hand, we can have a strategy with a longer expected time, but the probability that the search will end by a specific time is one. In addition, we prove optimality of strategies on the cube, octahedron, and dodecahedron.

In Chapter 4, we consider mixed strategies. Mixed strategies are a way to incorporate these asymmetric cases into a symmetric strategy. The way a mixed strategy works is that there is a set of n strategies that agents can choose from. All of the strategies are the same length. For each strategy i, there is an associated probability, p_i, of choosing that strategy. If the players do not see each other after running through their strategies, then they pick again with the same probabilities. In this chapter, we are primarily interested in finding the optimal expected time for the set of mixed strategies and determining whether there exists a mixed strategy that has a lower expected time than

our best known value. The calculations for mixed strategies are more complex. However, we present some simplifications that allow us to make significant progress on the previously mentioned goals.

Finally in Chapter 5, we conclude by summarizing our results alongside a discussion of optimality, interpretation of results, and directions for future research.

Chapter 2

Unbiased Random Walk Strategy

2.1 Unbiased Random Walk Strategy

An unbiased random walk strategy is the simplest strategy we consider. The two agents begin at vertices where they cannot see each other, and, at each iteration, move along a randomly chosen edge incident to their current vertex. By vertex transitivity, the agents have no preference for any direction, and thus even the first move is random. This strategy is equivalent to two simultaneous, unbiased, random walks on the edges of the platonic solid. The agents must move at each turn, and the search will end when the agents can see each other. Recall that we assume that the agents have full-face visibility, meaning the agents can see all faces (and their vertices) incident to their current vertex.

As a side note, we also have some results under adjacent-vertex visibility.

Under adjacent-vertex visibility, the agents can see only vertices adjacent to their current vertex. These results are presented in West [9]. One thing to note is that these detection ranges only differ on the cube and the dodecahedron, since these are the only solids whose faces are not triangular.

The unbiased random strategy provides upper bounds for the optimal expected time of a search. Thus, we will use the following results as comparisons going forward.

We will define $T(solid)$ to represent the expected time for the search to end on the given solid.

On all of the solids, calculations for this strategy can be done using first-step Markov chain decompositions. We explain the set up in an example calculation for the dodecahedron. However, almost all of the calculations can also be done using geometric probability mass functions, with the exception of the dodecahedron. To see more in depth explanations of the methods on the various solids, you can refer to [9] and [10].

2.1.1 Example Calculation

We present the expected time calculation for the unbiased random walk strategy on the dodecahedron.

On the dodecahedron, when the agents cannot see each other, they are either three, four, or five units apart.

Given that the agents are three units apart, there is a $\frac{4}{9}$ probability that they will end up three units apart again, a $\frac{1}{9}$ probability that they will end up four units apart, and a $\frac{1}{9}$ probability that they will end up five units apart. Thus, there is a $\frac{3}{9}$ probability that the agents will see each other after they take their step.

Given that the agents are four units apart, there is a $\frac{2}{9}$ probability that they will end up three units apart, a $\frac{5}{9}$ probability that they will end up four units apart again, and zero probability that they will end up five units apart. Thus, there is a $\frac{2}{9}$ probability that the agents will see each other.

Given that the agents are five units apart, there is a $\frac{2}{3}$ probability that they will end up three units apart, zero probability that they will end up four units apart, and a $\frac{1}{3}$ probability that they will end up five units apart again. Thus, there is zero probability that the agents will see each other.

Moreover, given that the agents cannot see each other, there is a $\frac{6}{10}$ probability that they are three units apart, a $\frac{3}{10}$ probability that they are four units apart, and a $\frac{1}{10}$ probability that they are five units apart.

Using this information, we set up the following system of equations.

Let E_i, for $i = 0, 1, 2, 3, 4, 5$, denote the expected time to end given the agents start i edges apart. Note that for $i = 0, 1, 2, E_i$ = 0 because the agents can already see each other when they are zero, one, or two units apart.

Let X_t denote minimum distance between the two agents at time t.

$$T(dodecahedron) = \frac{1}{10}E_5 + \frac{3}{10}E_4 + \frac{3}{5}E_3.$$

$$E_3 = 1 + \frac{1}{9}E_5 + \frac{1}{9}E_4 + \frac{4}{9}E_3.$$

$$E_4 = 1 + \frac{5}{9}E_4 + \frac{2}{9}E_3.$$

$$E_5 = 1 + \frac{1}{3}E_5 + \frac{2}{3}E_3.$$

The first equation for the overall expected time is an application of the law of total expectation.

The next three equations are applications of the following formula:

$$E_i = \sum_{k=0}^{5}(E_k + 1)P(X_1 = k \mid X_0 = i).$$

Note that we add one to the conditional expected time in the sum because the agents take one step.

Solving this system of equations, we get:

$$E_3 = \frac{51}{14}, \;\; E_4 = \frac{57}{14}, \;\; E_5 = \frac{36}{7}.$$

$$T(dodecahedron) = \frac{549}{140} \approx 3.9214.$$

2.1.2 Summary of Results

As mentioned before, we can use this set-up to calculate the expected time of the unbiased random strategy for all of the solids. Here, we summarize the results.

$$T(tetrahedron) = 0.$$

$$T(cube) = \frac{4}{3} \approx 1.3333.$$

$$T(octahedron) = \frac{3}{2} = 1.5.$$

$$T(dodecahedron) = \frac{549}{140} \approx 3.9214.$$

$$T(icosahedron) = \frac{5}{2} = 2.5.$$

Note that since we are working with face visibility, the game is trivial on the tetrahedron. On the tetrahedron, from every vertex you can see every other vertex.

We use these results as baseline results and preliminary upper bounds for

the optimal expected time for each solid. All further results are compared to these results.

Chapter 3

Pure Multi-Step Strategies

A multi-step strategy is a symmetric strategy where both agents first randomly choose a direction in which to move one step, and then move in a predetermined n-step sequence consisting of steps relative to the direction from which they came. These two parts combine to create an $n+1$-step strategy. In these strategies, all of the other classic search rules apply and the agents have full-face visibility. The search ends whenever the agents see each other, including if they see each other in the middle of their sequence of movements. If the agents do not see each other after completing the $n+1$ steps, then they repeat the process until they do see each other. We used first-step decomposition of absorbing Markov chains to calculate the expected meeting time of the strategies in this class.

Moreover, we consider right to be the complement of left, and thus we

say that two strategies are "reflections" of each other if each step besides the first random step are complements. For example, {random, left, right} and {random, right, left} are reflections. Because of the vertex-transitivity of the platonic solids, strategies that are reflections yield the same expected times. This reduces the number of strategies that we need to analyze by a factor of two.

3.1 The Two-Step Left Strategy

The simplest multi-step strategy we can construct is what we will call the Left Strategy. The agents each first randomly choose a direction in which to move one step, and then move one step in the left direction relative to the edge that they just came from. Recall that because of the vertex transitivity of the platonic solids, the expected times for reflections are the same. Thus, the Left Strategy will yield the same expected time as the respective Right Strategy. We denote the expected time of the Left Strategy as $T_L(solid)$, where the subscript L denotes the Left Strategy:

The icosahedron allows for more possible two-step strategies because it has the options of a hard left and a soft left (and respectively for the right). Thus, we have a Hard Left strategy where both agents first randomly choose a direction in which to move one step, and then move one step in the "hard left"

direction relative to the edge they just came from, and a respective Soft Left Strategy. This is illustrated in Figures 3.1 and 3.2. The notation is analogous to before, where the subscript HL refers to the Hard Left Strategy and SL refers to the Soft Left Strategy

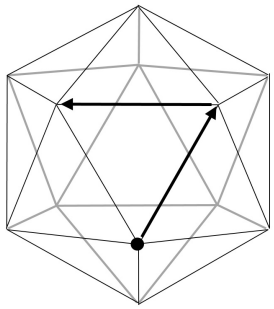

Figure 3.1: Illustration of a Hard Left on the icosahedron. The dot represents the starting position. The arrows denote the path of the agent. The first arrow corresponds to the first step where a direction is randomly chosen.

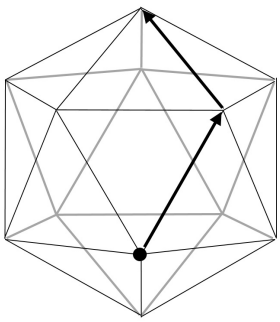

Figure 3.2: Illustration of a Soft Left on the icosahedron. The dot represents the starting position. The arrows denote the path of the agent. The first arrow corresponds to the first step where a direction is randomly chosen.

The calculations for these results are set up in a similar manner to the previous expected time calculations. We present the calculation for the expected time of the Hard Left Strategy on the icosahedron as an example. For the

icosahedron, the agents cannot see each other when they are either two units or three units apart.

Given that the agents are two units apart, there is a $\frac{4}{10}$ probability of seeing each other on the first step and a $\frac{7}{25}$ probability of seeing each other on the second step. Thus there is a $\frac{8}{25}$ probability that the agents will not see each other during the iteration of the sequence. More specifically, there is a $\frac{7}{25}$ probability that they will end up two units apart again and a $\frac{1}{25}$ probability that they will end up three units apart.

Given that the agents are three units apart, there is a $\frac{4}{10}$ probability of seeing each other on the first step and a $\frac{4}{10}$ probability of seeing each other on the second step. There is a $\frac{2}{10}$ probability that the agents will not see each other during the iteration of the sequence. More specifically, there is a $\frac{2}{10}$ probability that they will end up two units apart.

Moreover, given that the agents cannot see each other, there is a $\frac{5}{6}$ probability of being two units apart and a $\frac{1}{6}$ probability of being three units apart.

Using this information, we construct the following system of equations:

$$T_{HL}(icosahedron) = \frac{5}{6}E_2 + \frac{1}{6}E_3.$$

$$E_2 = \frac{4}{10}(1) + \frac{7}{25}(2) + \frac{7}{25}(E_2 + 2) + \frac{1}{25}(E_3 + 2).$$

$$E_3 = \frac{4}{10}(1) + \frac{2}{10}(2) + \frac{2}{10}(E_2 + 2).$$

Solving this system of equations yields the approximate expected time of 2.2921.

Performing these calculations on all of the solids yielded the following systems of equations and results.

Octahedron:

$$T_L(octahedron) = E_2.$$

$$E_2 = \frac{3}{4}(1) + \frac{1}{4}(2).$$

$$T_L(octahedron) = 1.25.$$

Cube:

$$T_L(cube) = E_3.$$

$$E_3 = \frac{2}{3}(1) + \frac{1}{3}(2).$$

$$T_L(cube) \approx 1.3333.$$

Dodecahedron:

$$T_L(dodecahedron) = \frac{6}{10}E_3 + \frac{3}{10}E_4 + \frac{1}{10}E_5.$$

$$E_3 = \frac{1}{3}(1) + \frac{2}{9}(2) + \frac{1}{3}(E_3 + 2) + \frac{1}{18}(E_4 + 2) + \frac{1}{18}(E_5 + 2).$$

$$E_4 = \frac{2}{9}(1) + \frac{1}{3}(2) + \frac{1}{9}(E_3 + 2) + \frac{2}{9}(E_4 + 2) + \frac{1}{9}(E_5 + 2).$$

$$E_5 = \frac{1}{3}(2) + \frac{1}{3}(E_3 + 2) + \frac{1}{3}(E_4 + 2).$$

$$T_L(dodecahedron) \approx 3.2859.$$

Icosahedron:

Hard Left Strategy:

For the system of equations, see the previous example.

$$T_{HL}(icosahedron) \approx 2.2921.$$

Soft Left Strategy:

$$T_{SL}(icosahedron) = \frac{5}{6}E_2 + \frac{1}{6}E_3.$$

$$E_2 = \frac{2}{5}(1) + \frac{6}{25}(2) + \frac{8}{25}(E_2 + 2) + \frac{1}{25}(E_3 + 2).$$

$$E_3 = \frac{2}{5}(1) + \frac{1}{5}(2) + \frac{1}{5}(E_2 + 2) + \frac{1}{5}(E_3 + 2).$$

$$T_{SL}(icosahedron) \approx 2.5273.$$

When comparing these results to our baseline results, we see that on all of the solids, the Left Strategy has a lower expected time.

3.2 Optimality of the Left Strategy on the Cube and Octahedron

On the cube and octahedron, the Left Strategy has the property that the agents are guaranteed to see one another by the end of the second step.

On the cube, the only positions where the agents cannot see each other are those that are three units apart. These vertices are on opposite sides of the solid. Since this is the only way for the agents to not see each other, the initial positions of the agents must be three units apart. An example of this can be seen in the Figure 3.1.

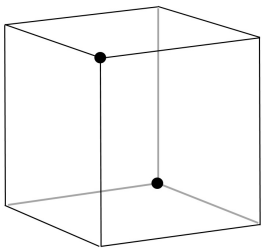

Figure 3.3: Initial positions on cube. The black dots denote the positions.

The Left Strategy starts with the agents randomly choosing a direction in which to move one step. There is a $\frac{2}{3}$ probability that the agents will see each other after the first step and a $\frac{1}{3}$ probability that the agents will not see each other after the first step. If agents do not see each other after the first step, it means that they moved to new vertices that are also three units apart. An illustration of this can be seen in Figure 3.2.

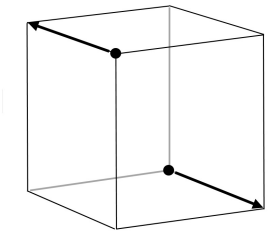

Figure 3.4: Positions after first step on cube. The black dots denote the initial positions. The bold arrows denote the paths of the two agents.

Assuming that the agents do not see each other after the first step, the agents then take one step in the left direction, relative to the direction from which they just came. On the cube, this step always results in the agents moving to vertices that are one unit apart, and thus the agents can see each other. An illustration of this can be seen in Figure 3.3.

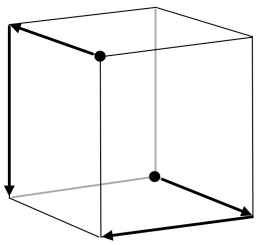

Figure 3.5: Final positions after one iteration of the Left Strategy on cube. The black dots represent the initial positions. The bold arrows denote the paths of the two agents.

The Left Strategy prevents the agents from executing an arbitrarily long sequence of movements where in each iteration, the agents end up three units apart. Part of the reason why the Left Strategy is so effective is because there is only one way (in terms of distance) in which the agents cannot see each other. This means that there is only one way in which the agents can start, so if we

know that they don't see each other, we can conclude how far apart they are.

The octahedron also shares many of these properties. On the octahedron, the agents cannot see each other only when they are two units apart. Therefore, the agents must start out two units apart. An example of this is shown in Figure 3.4.

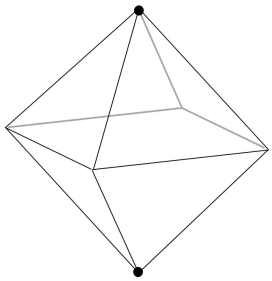

Figure 3.6: Initial positions on octahedron. The black dots denote the positions.

For the first step of the Left Strategy, there is a $\frac{3}{4}$ probability that the agents will see each other and a $\frac{1}{4}$ probability that they will not see each other after the first step. As discussed before, there is only one way for the agents to not see each other. Thus, if the agents do not see each other after the first step, it means they moved to new vertices that are also two units apart. An illustration of this is shown in Figure 3.5.

Assuming that the agents do not see each other after the first step, the agents will then move one step in the left direction. This will cause the agents to actually move to the same vertex and thus they meet. An illustration of this can be seen in Figure 3.6.

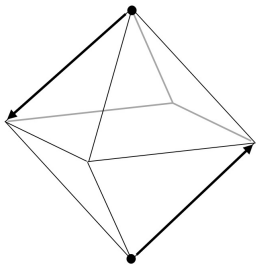

Figure 3.7: Positions after first step on octahedron. The black dots denote the initial positions. The bold arrows denote the paths of the two agents.

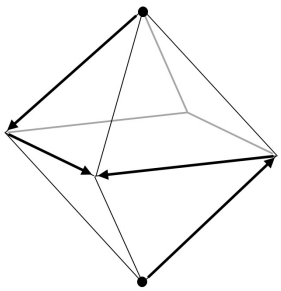

Figure 3.8: Final positions after one iteration of the Left Strategy on octahedron. The black dots denote the initial positions. The bold arrows denote the paths of the two agents.

A lot of the same principles that were discussed for the cube also apply to the octahedron. Thus, we reach the conclusion that for the cube and octahedron, the Left Strategy guarantees the agents see each other by the completion of the second step.

We are able to use this property to prove the optimality of this strategy out of all multi-step strategies. Before we prove optimality of the Left Strategy among all multi-step strategies, we first have the following lemma:

Lemma 1. *Let A be an n-step strategy that has a random first step and guarantees that the two agents see each other by the end of the n steps. Let B be a k-step strategy, $k > n$, such that the first $n - 1$ steps of the deterministic sequence are the same as those of A. Let $E_A(T)$ be the expected time for strategy A, and $E_B(T)$ be the expected time for strategy B. Then $E_A(T) = E_B(T)$.*

Proof. Let D_i denote the minimum distance between the two agents at time i and let S denote the set of distances where the agents can see each other. In other words, let S = $\{k$: when $D_i = k$, the agents can see each other$\}$

Given that the agents are using strategy A, consider a random variable M_A which denotes the time at which the agents see each other. In other words, $M_A =$ min{i: Under strategy A, $D_i \in S$}. The probability mass function of M_A, $P(M_A = m)$, describes the probability that the agents see each other at time m given they are using strategy A.

Since the agents will see each other once they reach the time M_A, the expected value of M_A describes the expected time at which the agents will see each other. Thus, $E_A(T)$ = $E(M_A)$

We are given that the n-step strategy, strategy A, guarantees the agents see each other. Therefore, we can reduce the possible values of M_A to 0,1,...n. We are also given that the first $n-1$ steps of the deterministic sequence in B are the same as those of A. Since A guarantees the agents see each other by the completion of the n^{th} step, B also has this guarantee. Therefore, we can conclude that the possible values for M_B are the same as M_A. In addition, the probability mass function of M_A is the same as that of M_B because the extra steps in B do not alter the probability of the agents seeing each other at any given time. Thus $E(M_A)$ = $E(M_B)$. Finally, we conclude $E_A(T)$ = $E_B(T)$. □

We will use this lemma to prove optimality of the Left Strategy on the cube and octahedron.

3.2.1 Optimality on the Cube

Let S_{cube} be the set of strategies which satisfy the following conditions:

1. The first step of the strategy is a random movement of one unit.

2. Both agents are always moving at a rate of one unit of length per unit of time. The agents never utilize a waiting strategy.

3. The strategy never intentionally retraces one or more of its previous steps.

For the cube, all of the n-step strategies, $n > 2$, in S_{cube} either start out with a random movement, then left movement or a random movement, then right movement. In other words, the first two steps of any strategy are the same as those of the Left Strategy (and its reflection, the Right Strategy). By the lemma, the expected value for any n-step strategies, $n > 2$, in S_{cube} is the same as that of the Left Strategy. Thus, the Left Strategy is optimal among all the n-step strategies, $n > 2$.

The only strategy in S_{cube} that is not an n-step strategy, $n > 2$, is the purely random strategy. In section 2, we calculated the expected value of this strategy, and yielded the result E(T) = 1.5. The expected time of $\frac{4}{3}$ for the Left Strategy is less than that of the random strategy. Thus, the Left Strategy is optimal out of all the strategies in the set S_{cube}.

3.2.2 Optimality on the Octahedron

Let S_{oct} be the set of strategies which satisfy the following conditions:

1. The first step of the strategy is a random movement of one unit.

2. Both agents are always moving at a rate of one unit of length per unit of time. The agents never utilize a waiting strategy.

3. The strategy never intentionally retraces one or more of its previous steps.

We will breakdown S_{oct} into 3 subsets.

Subset 1: For the octahedron, $\frac{2}{3}$ of the n-step strategies, $n > 2$, in S_{oct} either start out with a random movement, then left movement or a random movement, then right movement. In other words, the first two steps of the strategy are the same as those of the Left Strategy (and its reflection, the Right Strategy). By the previous lemma, this means that the expected value for any n-step strategy, $n > 2$, in S_{oct} is the same as that of the Left Strategy. Thus, the Left Strategy is optimal among this subset of strategies.

Subset 2: A subset of the remaining strategies will include either a left or right movement after one or more forward movements. For the octahedron, the only position where the agents cannot see each other is when they are on opposite nodes. This means that when the agents do not see each other after moving, their relative position remains the same. Given that the agents are on opposite nodes, a left movement (or right movement) guarantees that the agents will meet each other, while a forward movement guarantees that the agents will not see each other. Thus, given that the agents do not see each other after the random step, a strategy from this subset can be thought of as one random movement, $n - 3$ iterations of moving while maintaining their initial relative position, and then a Left Strategy (or Right Strategy) sequence.

Let A be an n-step strategy, $n > 1$, in S_{oct} whose first left or right movement occurs at the n^{th} step. Calculating the expected meeting time for this strategy,

we have

$$E_A(T) = (1)P(S_1(A)) + (n)P(S_n(A)) + \sum_{j=2}^{n-1} j(S_j(A)),$$

where S_i(A) denotes the event of the agents seeing each other on the i^{th} step using strategy A.

Assume the agents do not meet on the first step. Then for $j = 2, 3, ...n-1$, $P(S_j(A))$ = 0 because these are all forward movements. Also, since the n^{th} step is a left or right movement, the agents are guaranteed to see each at time n. Thus when calculating expected meeting time, we do not need to consider steps beyond the n^{th} step and the case where the agents do not see each other and need to retry the strategy. Substituting respective values, we get

$$E_A(T) = (1)\left(\frac{3}{4}\right) + (n)\left(\frac{1}{4}\right) + \sum_{j=2}^{n-1} j \times 0 = \frac{3+n}{4},$$

Note that as n increases, $E_A(T)$ increases. This means that all n-step strategies with $n > 2$ that guarantee the two agents see each other will have a longer expected time than the Left Strategy. Moreover, any strategies with an initial sequence consisting of one of these strategies will also have a longer expected time. Thus the Left Strategy is optimal among this subset of strategies.

Subset 3: The last subset of strategies are those that do not contain any left or right movements, only forward movements. Let A be an *n*-step strategy, $n > 1$, in S_{oct} that consists of the two agents moving forward for $n-1$ steps.

Then, the expected meeting time is

$$E_A(T) = (1)P(S_1(A)) + \sum_{j=2}^{n} j(S_j(A)) + P\left(\bigcap_{j=0}^{m} S_j^c(A)\right)(E_A(T) + n),$$

where S_i(A) denotes the event of the agents seeing each other on the i^{th} step using strategy A.

Since A consists of a sequence of only forward movements, for $j = 2, ...n$, $P(S_j(A))$=0. This means if the agents do not meet on the first step, they will not meet on the following $n-1$ steps, and will need to retry the sequence again.

Substituting respective values, we get

$$E_A(T) = (1)\left(\frac{3}{4}\right) + \sum_{j=2}^{n} j * 0 + \left(\frac{1}{4}\right)(E_A(T) + n).$$

This yields the following solution:

$$E_A(T) = 1 + \frac{n}{3}.$$

Note that as n increases, $E_A(T)$ increases. Thus the optimal strategy for this subset occurs for $n = 2$, yielding an expected time of $\frac{4}{3}$. This expected time is longer than that of the Left Strategy.

Thus, the Left Strategy is optimal out of all the strategies in set S_{oct}. An-

other result to note is that for the octahedron, face visibility and adjacent-node visibility are equivalent. Therefore, the Left Strategy is also optimal under adjacent-node visibility.

3.3 Longer Multi-Step Strategies

After investigating the two-step left strategy on the cube and octahedron, it is natural to examine longer multi-step strategies on the dodecahedron and icosahedron. The strategies consist of one random movement, and then varying sequences of left and right movements.

3.3.1 Dodecahedron

Let S_{dod} be the set of n-step strategies on the dodecahedron for $n = 1, 2, \ldots, 7$. We implemented the Markov chain approach in MATLAB and Python to analyze the expected times of the various strategies in S_{dod}. Through exhaustive search, we arrived at the following observations:

1. There exist 8 strategies, all of which are 7-step strategies, that guarantee that both agents see each other by the end of the sequence. Four of the strategies are the following:

   $$\{\text{random, left, left, right, left, left, right}\},$$

{random, left, left, right, right, right, left},

{random, left, right, right, left, right, right},

{random, left, right, right, right, left, left}

with the respective expected times 2.8, 2.9, 2.9667, 2.8667. Since opposite strategies yield the same results, the other four strategies with this property are the opposites of those listed above.

2. For the set S_{dod}, the optimal strategies are the following 7-step strategies: {random, left, left, right, left, left, right}, and its reflection. These strategies have an expected time of 2.8. Recall that these strategies guarantee that the agents see each other by the end of the sequence. It is important to note that while we have pinpointed the optimal pure multi-step strategy, we are unable to prove that it is optimal overall. However, it does provide a new upper bound on the overall optimal expected time.

3. The following strategies had expected times that are worse than the random strategy:

{random, left, right, left, right},

{random, left, right, left, right, left},

{random, left, right, left, right, left, right}

and their reflections, with the respective expected times 4.1875, 4.2638,

4.594. Recall that the expected time for the random strategy is approximately 3.9214. This result could be useful when considering "avoidance games", where the agents are trying to avoid each other.

3.3.2 Icosahedron

Again, the investigation of multi-step strategies on the icosahedron is distinct from those of the other solids. On the icosahedron, when constructing a predetermined sequence, we have the following choices: hard left, soft left, soft right, hard right. Let S_{ico} be the set of n-step strategies on the icosahedron for $n = 1, 2, 3, 4, 5$. We used Markov Chains to analyze the expected meeting times of the various strategies in S_{ico}. Through exhaustive search, we arrived at the following observations:

1. There exist twenty-four 5-step strategies which guarantee that both agents see each other by the end of the sequence. They are:

{random, hard left, soft left, soft left, soft left},
{random, hard left, soft left, hard right, soft right},
{random, hard left, soft right, hard left, soft right},
{random, hard left, soft right, hard right, soft left},
{random, hard left, hard right, hard left, soft left},

{random, hard left, hard right, soft right, soft right},

{random, soft left, soft left, hard left, hard right},

{random, soft left, soft left, soft left, hard left},

{random, soft left, hard left, hard right, hard left },

{random, soft left, hard left, soft right, hard right},

{random, soft left, hard right, soft left, hard right},

{random, soft left, hard right, soft right, hard left}

and their reflections, with respective expected times 2.1333 for the first six strategies and their opposites, and 2.2 for the remaining six strategies and their opposites.

2. In S_{ico}, the optimal strategies are the following sequences:

{random, hard left, soft left, soft left, soft left},

{random, hard left, soft left, hard right, soft right},

{random, hard left, soft right, hard left, soft right},

{random, hard left, soft right, hard right, soft left},

{random, hard left, hard right, hard left, soft left},

{random, hard left, hard right, soft right, soft right}

and their reflections, with respective expected time of approximately 2.1333. All of these strategies guarantee that the players see each other.

3. The strategy {random, hard left, hard left} has a worse expected time than the strategy containing its sub-sequence {random, hard left}. This is to be expected because the second movement in the hard left direction brings the agents to their original locations. Since the search starts in positions where the agents cannot see each other, going back to their original locations has no benefit in terms of expected time.

4. There are 152 multi-step strategies which have expected times that are worse than the random strategy. Recall that the expected time for the random strategy is 2.5.

3.4 Optimality on the Larger Solids

Using Lemma 1 from Section 2.2.2, we were able to prove optimality of the Left Strategy on the cube and the octahedron. It is natural to ask if we can adapt this proof to show that our optimal strategies of S_{dod} and S_{ico} are optimal overall. However, these larger solids are much more complicated.

There are certain properties that were integral to the optimality proof for the smaller solids. More specifically, for the octahedron, we relied on the fact that if the agents do not see each other after moving, it meant that the agents moved in the forward direction. We are then able to come to the conclusion that the expected time of a strategy is directly proportional to the number of steps

in it.

This is not the case on the dodecahedron. Firstly, for the dodecahedron, there are more ways for the agents to not see each other, i.e., they could be three units apart, four units apart, or five units apart. Additionally, there isn't a linear relationship between the expected time and the number of steps. For example, the strategy with the second best expected time is the following 6-step strategy: {random, left, left, right, left, left}. This 6-step strategy has a faster expected time than all of the 7-step strategies except for one (the optimal one). Another example involves an 8-step strategy: {random, left, left, left, right, left, left, right}. This strategy guarantees that the two agents see each other, while the corresponding 7-step strategy does not, and it has expected time of approximately 2.8333. This expected time is better than three of the 7-step strategies which have the same guarantee. Due to the non-uniqueness of these strategies and the varying expected times, it is hard to make any generalizing statements on the dodecahedron.

However, we can prove optimality of the strategy, {random, left, left, right, left, left, right} using other techniques. First, we must introduce the concept of calculating a lower bound for the expected time.

3.4.1 Lower Bound for Expected Time

In some cases, we may want to consider a lower bound on the expected time. The way this lower bound is calculated is we assume that if the players have not seen each other by the last step, then they see each other on the next step.

Here, we present an example calculation. Suppose that both agents use the Left Strategy. Recall that under the Left Strategy, the agent moves one step in a randomly chosen direction, and then one step in the left direction.

Given that the agents are three units apart, there is a $\frac{1}{3}$ probability that they will see each other on the first step and a $\frac{2}{9}$ probability that they will see each other on the second step. Thus, there is a $\frac{4}{9}$ probability that the agents do not see each other during the iteration of the sequence.

Given that the agents are four units apart, there is a $\frac{2}{9}$ probability that they will see each other on the first step and a $\frac{1}{3}$ probability that they will see each other on the second step. Thus, there is a $\frac{4}{9}$ probability that the agents will not see each other during the iteration of the sequence.

Given that the agents are five units apart, there is zero probability that the agents will see each other on the first step and a $\frac{2}{3}$ probability that they will see each other on the second step. Thus, there is a $\frac{1}{3}$ probability that the agents do not see each other during the iteration of the sequence.

Moreover, given that the agents cannot see each other, there is a $\frac{6}{10}$ probability of the agents being three units apart, a $\frac{3}{10}$ probability of the agents being

four units apart, and a $\frac{1}{10}$ probability of the agents being five units apart.

In this bounded expected time calculation, we assume that if the agents do not see each other during the iteration of the sequence, then they will see each other on the next step. Thus we assume the following:

Given that the agents start three units apart, there is a $\frac{4}{9}$ probability of the agents seeing each other on the third step. Given that the agents start four units apart, there is a $\frac{1}{3}$ probability of the agents seeing each other on the third step. Given that the agents start five units apart, there is a $\frac{2}{3}$ probability of the agents seeing each other on the third step.

Using the above information, we construct the following equations:

$$E_B(T) = \frac{6}{10}E_3 + \frac{3}{10}E_4 + \frac{1}{10}E_5$$

$$E_3 = \frac{1}{3}(1) + \frac{2}{9}(2) + \frac{4}{9}(3)$$

$$E_4 = \frac{2}{9}(1) + \frac{1}{3}(2) + \frac{4}{9}(3)$$

$$E_5 = \frac{1}{3}(2) + \frac{2}{3}(3)$$

Solving the equations and making the substitutions yields $E_B(T) = \frac{11}{5} = 2.2$ One thing to note is that for strategies that guarantee the two agents see each other by the end of the sequence, the true expected time is equal to the bounded expected time.

3.4.2 Proof of Optimality for the Dodecahedron

First, we consider the following lemma:

Lemma 2. *Let $E_B^M(T)$ be the bounded expected time calculated for a k-step pure multi-step strategy M, $k > 1$, $k \in \mathbb{Z}$. Let L_M denote the set of n-step strategies, $n \geq k$ such that the first k steps are the same as those of the strategy M. Then $E_B^M(T)$ is a lower bound for the expected times of the strategies in set L_M.*

Proof. $E_B^M(T)$ takes the following form: Let S_i be the event the agents see each other on step i. Note that the probability of not seeing each other by the k^{th} step is $P\left(\bigcap_{j=0}^{k} S_j^c\right)$, so

$$E_B^M(T) = \left[\sum_{i=1}^{k} iP(S_i)\right] + (k+1)P\left(\bigcap\nolimits_{j=0}^{k} S_j^c\right).$$

For every n-step strategy in L_M, we can calculate a bounded expected time, $E_B^{(n)}(T)$, that takes the following form:

$$E_B^{(n)}(T) = \left[\sum_{j=1}^{n} jP(S_j)\right] + (n+1)P\left(\bigcap\nolimits_{j=0}^{n} S_j^c\right).$$

Since the first k steps of the strategy are the same, we can write $E_B^{(n)}(T)$ as

$$E_B^{(n)}(T) = \left[E_B^M(T) - (k+1)P\left(\bigcap\nolimits_{j=0}^{k} S_j^c\right)\right] + \left[\sum_{j=k+1}^{n} jP(S_j)\right] + (n+1)P\left(\bigcap\nolimits_{j=0}^{n} S_j^c\right).$$

For the n-step strategy, each event S_j for $j \geq k+1$ is a subset of the event $\left(\bigcap_{j=0}^{k} S_j^c\right)$. The event $\left(\bigcap_{j=0}^{n} S_j^c\right)$ is also a subset of the event $\left(\bigcap_{j=0}^{k} S_j^c\right)$. Together, the events S_j for $j \geq k+1$ and $\left(\bigcap_{j=0}^{n} S_j^c\right)$ create a partition of the event $\left(\bigcap_{j=0}^{k} S_j^c\right)$. Thus

$$\sum_{j=k+1}^{n} P(S_j) + P\left(\bigcap_{j=0}^{n} S_j^c\right) = P\left(\bigcap_{j=0}^{k} S_j^c\right).$$

Using this information we conclude that

$$\sum_{j=k+1}^{n} jP(S_j) + (n+1)P\left(\bigcap_{j=0}^{n} S_j^c\right) \geq (k+1)P\left(\bigcap_{j=0}^{k} S_j^c\right).$$

Thus, $E_B^{(n)}(T) \geq E_B^M(T)$.

Let $E^{(n)}(T)$ denote the expected time for a respective n-step strategy. We can conclude that for all n-step strategies, $E^{(n)}(T) \geq E_B^M(T)$. □

In order to prove the strategy {random, left, left, right, left, left, right} is optimal, we apply the bounded expected time calculation to all the pure strategies that do not guarantee the two agents do not see each other.

We find that all of the bounded expected times are greater than 2.8. By Lemma 2, these lower bounds are also lower bounds for the expected times of n-step strategies, $n \geq 7$. Thus, all n-step strategies have expected times that are greater than 2.8. Therefore, 2.8 is the optimal expected time for the set of pure multi-step strategies on the dodecahedron.

3.5 Observations

One of the benefits of working with all of the platonic solids is that they have a varying set of properties. For example, when comparing the icosahedron to the dodecahedron, the icosahedron has a larger number of faces than the dodecahedron, but it also has fewer nodes and the degree of each node is higher. Thus, analyzing patterns that occur across all the solids can give some insight into what types of strategies are better.

In investigating the multi-step strategies on the various platonic solids, one interesting pattern has appeared in all of them. In all of the respective optimal strategies for the solids, the agents move in such a way where they trace the perimeter of their original area of visibility for a certain period of time before moving onto a completely new area of visibility. This is illustrated in Figures 3.7, 3.8, 3.9, and 3.10. In addition, strategies that are constantly going to new areas of visibility do worse than the simple random strategy.

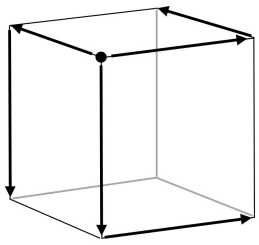

Figure 3.9: Illustration of the Left Strategy on cube. The black dot represents the initial position. The bold arrows represent possible paths of the agent. All three possible paths are shown.

It appears to be advantageous for an agent to not completely abandon their

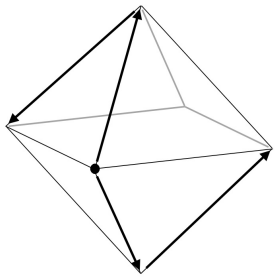

Figure 3.10: Illustration of the Left Strategy on the octahedron. The black dot represents the initial position. The bold arrows represent possible paths of the agent. Two out of four possible paths are shown.

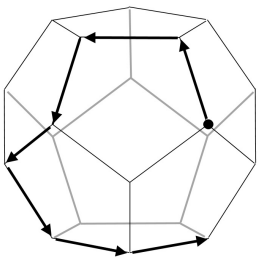

Figure 3.11: {random, left, left, right, left, left, right} strategy on dodecaedron. The black dot represents the initial position. The bold arrows represent the path of the agent. For simplicity, only one possible path is shown.

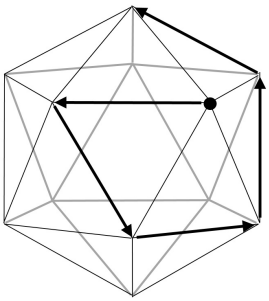

Figure 3.12: {random, hard left, soft left, soft left, soft left} strategy on icosahedron. The black dot represents the initial position. The bold arrows represent the path of the agent. For simplicity, only one possible path is shown.

original range of sight. Perhaps the intuition for this is behind the optimal asymmetric strategy, the "Wait for Mommy" strategy where one players waits and the other visits all the possible locations. The optimal strategies are an interesting mix of waiting by staying close to your original location, but also moving and gaining new sight. A possible reason for why constantly going to new areas of visibility yields such poor results is that the players are more likely to move around each other, resulting in them not seeing each other.

In adapting this to the sphere, we assume that the players' area of visibility is half of the sphere. We consider a strategy where the players travel r length units to the perimeter of their original area of visibility, travel along that perimeter for r length units in some specified direction. Then if needed, the players follow a sequence of turns, traveling for r length units each turn. The number of steps needed in the sequence and the degree at which to turn is likely dependent on how long the distance r is. One direction for future research is investigate how effective this class of strategies is on the sphere.

Chapter 4

Mixed Strategies

After looking at multi-step strategies, we were interested in expanding the strategy set to mixed strategies. Previous work and even our own numerical results suggest that asymmetric cases yield lower expected times than symmetric cases. Mixed strategies are a way to incorporate these asymmetric cases into a symmetric strategy. The way a mixed strategy works is that there is a set of n strategies that agents can choose from. All of the strategies are the same length. For each strategy i, there is an associated probability of choosing that strategy, p_i. If the players do not see each other after running through their strategies, then they can pick again with the same probabilities. The pure strategies are a subset of the mixed strategies where one probability is set equal to 1 and the others to 0.

4.1 Lower Bound for Expected Time

Allowing the players the option to choose again complicates the calculations much more. Therefore, in order to simplify the computations, we consider the lower bound on the expected time that we calculated in section 3.4.1. Recall that the way this lower bound is calculated is we assume that if the players have not seen each other by the last step, then they see each other on the next step. One thing to note is for strategies that guarantee the two agents see each other by the end of the sequence, the true expected time is equal to the bounded expected time.

Let $\mathcal{S} = \{S_1, S_2, ...S_n\}$ be a set of n pure strategies. Suppose the agents use a mixed strategy where they choose their strategies from set $\mathcal{S}$. Let B_{ij} denote the bounded expected time given that agent 1 uses strategy i and agent 2 uses strategy j. Note that $B_{ij} = B_{ji}$. After calculating all of the B_{ij} terms, to calculate a lower bound for the overall expected time of the mixed strategy, we need to optimize with regards to the constraints and objective function listed below:

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} p_i p_j B_{ij}.$$

$$s.t. \sum_{k=1}^{n} p_k = 1.$$

$$\forall k, p_k \geq 0.$$

The objective function is an application of the Law of Total Expectation. Each term in the sum represents the probability that agent 1 uses strategy i and agent 2 uses strategy j, multiplied by the bounded expected time, given that strategies i and j are used. The constraints ensure that our optimal p_i values satisfy the probability axioms.

To gain some intuition for the new system stated in the previous section, first consider a case where the agents choose their strategy at the beginning and they stick with that strategy in every iteration until they see each other. In other words, before moving the agents pick their strategy. They go through one iteration of that strategy, and if they do not see each other they repeat the same strategy. There is no option to re-choose. Therefore, we can partition the expected time by what strategies are chosen. Let E_{11} be the expected time given both agents choose strategy 1. Let E_{22} be the expected time given both agents choose strategy 2. Let E_{12} be the expected time given the agents choose different strategies (Who chooses which strategy does not matter since the solids are vertex-transitive and the agents are indistinguishable.) Suppose that p is the probability of picking strategy 1, and thus $1-p$ is the probability of picking strategy 2. Thus, there is a p^2 probability of both agents choosing strategy 1, a $2p(1-p)$ probability of the agents choosing different strategies, and a $(1-p)^2$ probability of both agents choosing strategy 2. Using the law of total expectation, we have the following expression: $E(T) = p^2E_{11} + 2p(1-p)E_{12} + (1-p)^2E_{22}$.

However, the problem we presented allows the agents to re-choose strategies at every iteration. Therefore, the expression we have does not accurately solve for the expected time that we are looking for. Our way around that is to replace E_{ij} with our bounded expected times (method and example presented in the beginning of Chapter 4.), resulting in a lower bound for the expected time of the mixed strategy. Let B_{ij} denote the bounded expected time given that agent 1 uses strategy i and agent 2 uses strategy j. Note that $B_{ij} = B_{ji}$. When mixing strategy i and strategy j, we have the following equation for the bounded expected time (we say bounded because this calculation is done using our lower bounds): $E_B(T) = p^2B_{11} + 2p(1-p)B_{12} + (1-p)^2B_{22}$. To find the bounded expected time, we minimize this expression with respect to p, given the constraint that p is between 0 and 1, inclusive. We consider $p \in arg\,min\{E_B(T) : 0 \leq p \leq 1\}$ to be the associated optimal probability

Overall, we are still able to make some claims using these lower bounds on expected time. Specifically for the case when we are mixing two strategies, there are some interesting results that relate the overall expected time to the expected time of specific cases.

4.2 Reducing the Number of Strategies

While we don't have specific conditions for general n, we still do have some results that allow us to narrow our region of focus. We present two lemmas which reduce the number of mixed strategies that we need to analyze, given that the goal is to find a strategy with a lower expected time.

Lemma 3. *Let $E^*(T)$ be the optimal time for the set of pure strategies. Let $\{S_1, S_2, \ldots S_n\}$ be a set of n pure strategies. If for all $i, j, B_{ij} \geq E^*(T)$, then for any associated mixed strategy M, $E_M(T) \geq E^*(T)$, where $E_M(T)$ is the expected time of the strategy M.*

Proof. Let p_i denote the probability of choosing S_i for $i = 1, 2, \ldots n$ such that $\sum_{i=1}^{n} p_i = 1$ and for all i, $p_i > 0$. For the expected time of the mixed strategy, we have the following inequality:

$$E(T) \geq \sum_{i=1}^{n} \sum_{j=1}^{n} p_i p_j B_{ij}$$

Recall that for the right hand side, this is our bounded expected time for the mixed strategy.

We are given in the lemma that for all i, j, $B_{ij} \geq E^*(T)$.

Thus we can write

$$E(T) \geq \sum_{i=1}^{n}\sum_{j=1}^{n} p_i p_j B_{ij} \geq \sum_{i=1}^{n}\sum_{j=1}^{n} p_i p_j E^*(T)$$

We can then simplify the last expression with the following steps:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} p_i p_j E^*(T) = E^*(T)\sum_{i=1}^{n}\sum_{j=1}^{n} p_i p_j = E^*(T)\left(\sum_{i=1}^{n} p_i\right)^2 = E^*(T).$$

Thus, $E(T) \geq E^*(T)$. □

From this lemma, we can conclude that given for all $i, j, B_{ij} \geq E^*(T)$, if there exists i, j such that $B_{ij} > E^*(T)$, then the inequality becomes a strict inequality, meaning the strategy is not optimal.

Also when n = 2, Lemma 1 is a statement about what pairs of pure strategies create mixed strategies that perform better. More specifically, the only mixed strategies that perform better than pure strategies are those with $B_{12} < E^*(T)$.

Lemma 4. *Let S = $\{S_1, S_2, \ldots\ S_n\}$ be a set of n pure strategies and M be the mixed strategy that consists of the n strategies in the set. Suppose there exists some U such that $E_M(T) < U$ and for some strategy S_i, $B_{ij} \geq U$ (and $B_{ji} \geq U$) for j = 1,2,...n. Let $\hat{M}$ be the mixed strategy consisting of the strategies $\{S_1, S_2, \ldots S_{i-1}, S_{i+1}, \ldots S_n\}$. Then $E_{\hat{M}}(T) < E_M(T)$*

Proof. Let X: $S \times S \to \mathbb{R}$ be a random variable denoting the bounded expected time corresponding to the pair of randomly chosen strategies S_i and S_j respectively. Let A be the event that S_i is used by at least one of the agents. We will only consider the nontrivial case where $P(A) \neq 0$.

The possible values for our random variable X are B_{ij} for all combinations of i and j. The probability that X = B_{ij} for a given i, j is equal to the probability of agent 1 choosing strategy S_i and agent 2 choosing strategy S_j. Thus, the expected value of X is the bounded expected time for the mixed strategy. i.e. $E_M(T) = E(X)$.

We want to partition the expected value of X using the event A. Using the law of total expectation, we have the following expression:

$$E(X) = P(A)E(X \mid A) + (1 - P(A))E(X \mid A^c)$$

.

It is given in the lemma that for S_i, $B_{ij} \geq U \; \forall j$. From there, we can conclude that this means $E(X \mid A) \geq U$. Since $E(X) = E_M(T) < U$, we make the following claim: $E(X \mid A^c) < E(X)$. We will now prove this claim by way on contradiction.

Claim: $E(X \mid A^c) < E(X)$.

Proof. For sake of contradiction, assume that $E(X \mid A^c) \geq E(X)$.

We have $E(X) = E(X \mid A)P(A) + E(X \mid A^c)P(A^c)$.

We concluded previously that $E(X \mid A) > U$. Since $E(X) < U$, $E(X \mid A) > E(X)$. Using these inequalities, we can write

$$E(X) = E(X \mid A)P(A) + E(X \mid A^c)P(A^c) > E(X)P(A) + E(X)P(A^c).$$

Simplifying the expression, we get $E(X) > E(X)(P(A) + P(A^c) = E(X)$.

We arrive at a contradiction. □

Suppose that we now take S_i out of the set of pure strategies from which the agents can choose. Therefore, the probability of choosing strategy S_i is now zero. We set $P(A) = 0$ (in other words, we set $P(X = B_{kj}) = 0$ for $k = i$ or $j = i$). Also, for the pairs of strategies S_k and S_j in A^c we set $P(X = B_{kj}) = P(X = B_{kj} \mid A^c)$.

Then $E_{new}(X) = E(X \mid A^c)$. Recall $E(X \mid A^c) < E_M(T)$. Thus $E_{new}(X) < E_M(T)$.

Let $E_{\hat{M}}(T)$ denote the optimal expected time with respect to the vector of probabilities. We know that $E_{\hat{M}}(T) \leq E_{new}(X)$ because the probabilities associated with $E_{new}(X)$ do not necessarily minimize the expected time.

Thus, we have the following string of inequalities: $E_{\hat{M}}(T) < E_{new}(X) < E_M(T)$. □

4.3 Results for the Dodecahedron

4.3.1 Mixing Two Strategies on the Dodecahedron

So far, we have only looked at mixing two 7-step strategies on the dodecahedron. Using the previous lemmas, we have been able to reduce the number of strategies that we need to analyze. There are sixty-four 7-step strategies, meaning that besides pure strategies, there are $\binom{64}{2}$ = 2016 mixed strategies.

Using Lemma 3, we were able to reduce our set of strategies to 162 strategies. Since we are only mixing two strategies, Lemma 4 is unable to further reduce our set. When only mixing two strategies, Lemma 4 states that if there exists some U such that $E_M(T) < U$ and for some strategy $S_i, B_{ij} \geq U$ for $j = 1, 2$, then for the pure strategy that is not S_i, the expected time would be less than that of the mixed strategy.

In Lemma 3, we have restricted $B_{12} < 2.8$. Moreover, through exhaustive search we know that for all pure strategies, $B_{ii} \geq 2.8$. Therefore, $B_{11} \geq 2.8$ and $B_{22} \geq 2.8$, regardless of what strategies they are. Thus, $E_M(T) \geq B_{12}$. Since $E_M(T) \geq B_{12}$, there does not exist a U such that $E_M(T) < U$, $B_{12} \geq U$ and either $B_{11} \geq U$ or $B_{22} \geq U$. Thus there are no further reductions from Lemma 4.

In addition, we have the following lemma.

Lemma 5. *For mixed strategies on the dodecahedron, if $B_{ij} < 2.8$, then it is guaranteed that the absolute minimum of the expected time function is attained and the optimal probability, p, will be between 0 and 1, inclusive.*

Proof. Let $E_B(T)$ denote the bounded expected time for the mixed strategy. Recall that we have the following expression:

$$E_B(T) = p^2 B_{11} + 2p(1-p)B_{12} + (1-p)^2 B_{22}.$$

Expanding this expression, we get the following:

$$(B_{11} - 2B_{12} + B_{22})p^2 + (2B_{12} - 2B_{22})p + B_{22}.$$

For any parabola of the form $f(x) = ax^2 + bx + c$, where a, b, c are real constants, the parabola has a finite minimum only when a is strictly greater than zero. (When a is strictly greater than zero, the parabola is convex.) Therefore, the bounded expected time function has a finite minimum only when $B_{11} - 2B_{12} + B_{22} > 0$. This is what we will attempt to prove.

Given $B_{12} < 2.8$ and for all i, $B_{ii} \geq 2.8$, we conclude $B_{11} + B_{22} \geq 2 * 2.8$ and $2B_{12} < 2 * 2.8$. Thus $B_{11} + B_{22} > 2B_{12}$. (i.e. $B_{11} - 2B_{12} + B_{22} > 0$). Therefore, the

function has a finite minimum.

Moreover, for any parabola of the form $f(x) = ax^2 + bx + c$, where $a > 0$, b ,c are real constants, the vertex of the parabola (the x value associated to the minimum functional value) is $\frac{-b}{2a}$. For the bounded expected time function, the vertex is equal to

$$\frac{-2(B_{12} - B_{22})}{2(B_{11} - 2B_{12} + B_{22})}.$$

We can write $(B_{11}-2B_{12}+B_{22})$ as $-(B_{12}-B_{22})+(B_{11}-B_{12})$. Because $B_{11} \geq 2.8$ and $B_{12} < 2.8$, $(B_{11} - B_{12}) > 0$.

Thus $-(B_{12} - B_{22}) + (B_{11} - B_{12}) > -(B_{12} - B_{22})$ and

$$\frac{-(B_{12} - B_{22})}{(B_{11} - 2B_{12} + B_{22})} < 1.$$

Furthermore, $-(B_{12} - B_{22}) > 0$ because $B_{12} < 2.8$ and $B_{22} \geq 2.8$, and $(B_{11} - 2B_{12} + B_{22}) > 0$ because $-(B_{12} - B_{22}) > 0$ and $(B_{11} - B_{12}) > 0$. Thus

$$\frac{-(B_{12} - B_{22})}{(B_{11} - 2B_{12} + B_{22})} > 0.$$

Therefore, the p-coordinate of the vertex is between 0 and 1 inclusive.

□

This result reaffirms the potential for mixed strategies to yield lower ex-

pected times than pure strategies. In addition, we can come up with a set of conditions for when it is better to use a pure strategy versus a mixed strategy. For example, when $B_{11} - 2B_{12} + B_{22} \leq 0$, the function goes to negative infinity. Thus, the p value corresponding to the minimized bounded expected time will be at the boundary points: p = 0 or p = 1. Moreover, if the vertex of the graph is outside of the interval [0,1], the p value will be at the boundary points.

4.3.2 Bounds for the Optimal Expected Time

Our ultimate goal is to find the optimal expected time for the set of mixed strategies. Since we are not calculating exact expected times, performing exhaustive search on current numerical results will not give an answer to this question. However, upper and lower bounds to this number can be found.

An upper bound of the optimal expected time is the expected time of our optimal pure strategy, 2.8. We find an improved lower bound by finding the lowest of the lower bound expected times.

To find the lower bound, you could use exhaustive search. However, we use the bounded expected times of the asymmetric cases to partition the strategy set. We then utilize the following lemmas to gradually increase and check the lower bound until we cannot further increase it.

Lemma 6. *Let $E^*(T)$ denote the optimal expected time for the set of mixed strate-*

gies. Let B_{12}^L denote the smallest B_{ij} value for $i \neq j$. Let B_{11}^L denote the smallest B_{ii} value and B_{22}^L denote the second smallest B_{ii} value (B^L11 can equal B_{22}^L if more than one pure strategy corresponds to the lowest B_{ii} value.) Let $E_{opti}(T)$ be the minimized value of $p^2B_{11}^L + 2p(1-p)B_{12}^L + (1-p)^2B_{22}^L$ with respect to p. $E_{opti}(T) < E^*(T)$.

Proof. $E^*(T) = p^2B_{11} + 2p(1-p)B_{12} + (1-p)^2B_{22}$ for some two strategies and some optimal value p. Let p^* denote the corresponding optimal value of p for $E^*(T)$. Also, let $B_{11}^*, B_{12}^*, and B_{22}^*$ be the corresponding B_{ij} values for $E^*(T)$. So we have $E^*(T) = (p^*)^2B_{11}^* + 2p^*(1-p^*)B_{12}^* + (1-p^*)^2B_{22}^*$.

We know that $B_{11}^L > 0$ is the smallest B_{ii} value. Thus, $B_{11}^L \leq B_{11}^*$. Suppose we replace B_{11}^* with B_{11}^L in the above expression. We get the following inequality: $E^*(T) \geq (p^*)^2B_{11}^L + 2p^*(1-p^*)B_{12}^* + (1-p^*)^2B_{22}^*$.

Using the same logic, we can make the analogous replacements for B_{12}^* and B_{22}^*. Then we get the inequality $E^*(T) \geq (p^*)^2B_{11}^L + 2p^*(1-p^*)B_{12}^L + (1-p^*)^2B_{22}^L$.

$E_{opti}(T)$ is the minimized value of $p^2B_{11}^L + 2p(1-p)B_{12}^L + (1-p)^2B_{22}^L$ with respect to p. So $E_{opti}(T) \leq (p^*)^2B_{11}^L + 2p^*(1-p^*)B_{12}^L + (1-p^*)^2B_{22}^L$.

We can then create the following string of inequalities:

$$E_{opti}(T) < (p^*)^2B_{11}^L + 2p^*(1-p^*)B_{12}^L + (1-p^*)^2B_{22}^L \leq E^*(T).$$

Thus $E_{opti}(T)$ is a lower bound for the true optimal expected value. □

Lemma 7. *Let $T(\gamma)$ denote the set of bounded expected times for a mixed strategy such that $B_{12} = \gamma$ for a given $\gamma > 0$. Let $E_k(T)$ denote the minimized value of $p^2 B_{11}^L + 2p(1-p)k + (1-p)^2 B_{22}^L$ with respect to p. Then $E_k(T)$ is less than or equal to the minimum value in the set $T(\gamma)$.*

Proof. Let $E_{min}(T)$ denote the minimum value in the set $T(\gamma)$. With this value, there is an associated mixed strategy, and thus an associated B_{11} and B_{22} value. Suppose $B_{11} \leq B_{22}$. Let's call these values B_{11}^{min} and B_{22}^{min} respectively. Also associated with $E_{min}(T)$ is the optimal p value, which we will call P^*.

Recall that B_{11}^L and B_{22}^L are the two smallest B_{ii} values overall. Thus, $B_{11}^L \leq B_{11}^{min}$ and $B_{22}^L \leq B_{22}^{min}$.

Thus we can conclude

$$(p^*)^2 B_{11}^L + 2p^*(1-p^*)\gamma + (1-p^*)^2 B_{22}^L < (p^*)^2 B_{11}^{min} + 2p^*(1-p^*)\gamma + (1-p^*)^2 B_{22}^{min}.$$

In the lemma, $E_\gamma(T)$ is defined as the minimized value of $p^2 B_{11}^L + 2p(1-p)\gamma + (1-p)^2 B_{22}^L$ with respect to p. So $E_\gamma(T) \leq (p^*)^2 B_{11}^L + 2p^*(1-p^*)\gamma + (1-p^*)^2 B_{22}^L$.

By combining these inequalities, we reach our final result: $E_\gamma(T) \leq E_{min}(T)$.

□

Lemma 8. *Let $E(\gamma)$ denote the minimized value of $p^2 B_{11}^L + 2p(1-p)\gamma + (1-p)^2 B_{22}^L$ with respect to p. Suppose $\gamma_1 < \gamma_2$. Then $E\gamma_1) \leq E(\gamma_2)$.*

Proof. For sake of contradiction, suppose that $E(\gamma_1) > E(\gamma_2)$. Let p^* denote the

associated optimal value of p for $E(\gamma_1)$ and p^{**} denote the associated optimal value of p for $E_(\gamma_2)$.

Since $E(\gamma_2) < E(\gamma_1)$, this means that

$$(p^{**})^2 B_{11}^L + 2p^{**}(1-p^{**})\gamma_2 + (1-p^{**})^2 B_{22}^L < (p^*)^2 B_{11}^L + 2p^*(1-p^*)\gamma_1 + (1-p^*)^2 B_{22}^L.$$

Since $0 < \gamma_1 < \gamma_2$, we can conclude that

$$(p^{**})^2 B_{11}^L + 2p^{**}(1-p^{**})\gamma_1 + (1-p^{**})^2 B_{22}^L < (p^{**})^2 B_{11}^L + 2p^{**}(1-p^{**})\gamma_2 + (1-p^{**})^2 B_{22}^L.$$

Stringing these inequalities together, we get that

$$(p^{**})^2 B_{11}^L + 2p^{**}(1-p^{**})\gamma_1 + (1-p^{**})^2 B_{22}^L < (p^*)^2 B_{11}^L + 2p^*(1-p^*)\gamma_1 + (1-p^*)^2 B_{22}^L.$$

In other words,

$$(p^{**})^2 B_{11}^L + 2p^{**}(1-p^{**})\gamma_1 + (1-p^{**})^2 B_{22}^L < E(\gamma_1).$$

However, $E(\gamma_1)$ is supposed to be the minimized value of $p^2 B_{11}^L + 2p(1-p)\gamma_1 + (1-p)^2 B_{22}^L$ with respect to p. We arrive at a contradiction. □

For the dodecahedron, when optimizing $p^2 B_{11} + 2p(1-p)B_{12} + (1-p)^2 B_{22}$ with respect to p, we use the lower bound value 2.6333 for B_{12}. This value was

found through exhaustive search. For B_{11} and B_{22} we use the two lowest lower bounds for 7-step pure strategies, which are 2.8 and 2.8. This calculation yields our initial lower bound of 2.7165.

From Lemma 7, we know that for all the mixed strategies with B_{12} = 2.6333, the bounded expected times are greater than or equal to 2.7165. After analyzing the mixed strategies that satisfy B_{12} = 2.63333, we find that none of the pure strategy combinations have expected times 2.8 and 2.8. This means the bounded lower bound is never attained, and thus there is room for improvement. Looking at the set of mixed strategies with B_{12} = 2.6333, the minimum value is 2.7537037. Consider this value to be our new tentative lower bound.

Next, consider the second lowest B_{ij} value, 2.6667. Optimizing the expected time (still using 2.8 and 2.8 as the expected times for the pure strategies), yields the value 2.733345. From Lemma 5, we conclude that all of the mixed strategies in this set have bounded expected times greater than or equal to 2.733345. Analyzing the subset of strategies where B_{12} = 2.6667, we find that none of the mixed strategies have a combination of pure strategies with expected times 2.8 and 2.8, and the minimum value is 2.75897436. This value is greater than the tentative lower bound, 2.7537037. Therefore, the tentative lower bound holds.

This process can be repeated until we reach a B_{ij} value, B^*_{ij}, such that the optimized expected time is greater than the lower bound. Lemma 8 states that all strategies with $B_{ij} > B^*_{ij}$ will have a larger expected time.

After executing this process, the final result for the lower bound is 2.7537037. Therefore our current bounds for the optimal expected time for the set of mixed strategies are 2.7537037 and 2.8. This interval is relatively small, with an approximate length of 0.05.

4.3.3 Conjecture

While we have not been able to calculate the optimal expected time of mixed strategies, I conjecture that there does exist a mixed strategy with an expected time strictly less than 2.8.

The mixed strategy associated with the lower bound of our interval is composed of the following pure strategies: {random, left, left, left, left, left, right}, {random, left, left, right, left, left, right}. Recall that this strategy had a bounded expected time of approximately 2.7537. For this mixed strategy, there are two cases where the agents are guaranteed to see each other by the completion of the seventh step, when both agents choose {random, left, left, right, left, left, right} or when the agents choose different strategies. Thus, the case of most interest to us is when the agents both use {random, left, left, left, left, left, right}.

Consider a variation of a mixed strategy where instead of having the opportunity to re-pick after completing each sequence of steps, the agents choose their strategies in the beginning and stick with the same strategy. Optimiz-

ing with respect to these conditions, we find that the strategy composed of {random, left, left, left, left, left, right} and {random, left, left, right, left, left, right} yields an expected time of approximately 2.7680. This value is strictly less than 2.8, the optimal expected time for pure strategies. So we can conclude that there does exist a strategy (however, not from the specific class we are considering) that performs better than all of the pure multi-step strategies.

Furthermore, I believe that the expected time of this variant should be an upper bound for the expected time of the mixed strategy composed of the strategies, {random, left, left, left, left, left, right} and {left, left, right, left, left, right}. As mentioned previously, the main case of interest is when both agents use the strategy {random, left, left, left, left, left, right}. For the variant, we assume that if both agents use {random, left, left, left, left, left, right}, then they will continue to use it until they see each other. The mixed strategy allows the opportunity to re-choose. A "worst-case" scenario would be that both agents consistently choose {random, left, left, left, left, left, right} and continue to not see each other for an arbitrarily long amount of time. Therefore, I conjecture that the expected time of the mixed strategy composed of the strategies, {random, left, left, left, left, left, right} and {random, left, left, right, left, left, right} is between 2.7537 and 2.7680.

If this conjecture can be proven, then our results support the incorporation of asymmetric cases as a way to decrease expected time. Of course, whether

this difference is significant is a different matter to discuss.

Chapter 5

Summary and Conclusion

5.1 Summary of Results

We have investigated expected meeting times for our simplified version of the astronaut problem on the platonic solids. We explored random walks, multi-step strategies, and mixed strategies. These results are the first of their kind in modeling the expected time for two astronauts to see each other in the astronaut problem. Overall, there are some more intuitive results that are consistent with the current literature, and there are more surprising results that illustrate thought-provoking ideas in the problem.

We first considered an unbiased random walk strategy. We considered this to be the most basic strategy since the agents are not trying to strategize in any way. These baseline cases are used as comparisons for other strategies

as we attempt to decrease the current optimal expected meeting time. One intuitive result is that the unbiased random walk yields expected times that get longer as the solids get bigger. This makes sense because as the solids get larger, there are more edges to traverse and the proportion of nodes that can be seen decreases. More work on the unbiased random walk strategy can be found in [9], which also explores how changes in visibility affects the expected time.

The Left Strategy serves as an introduction to multi-step strategies. It gives the agents a path that is not fully random. It decreases the expected time on the three solids examined – the octahedron, cube, and dodecahedron under face visibility. This verifies that non-random, multi-step strategies can be better than random strategies. In fact, we prove that the Left Strategy is optimal on the octahedron and cube due to the fact that it guarantees the agents see each other by the end of the second step. It is important to note that the expected time for the Left Strategy on these smaller solids is equal to the expected time of the random strategy with the optimal waiting probability. However, the Left Strategy achieves a different criterion for optimality since it guarantees that the agents see each other by the end of the second step, while the random walk strategy does not.

The Soft/Hard Left Strategy on the icosahedron also produced some interesting results. Both of these strategies yielded expected times that are shorter

than the expected time of the random strategy, $\frac{5}{2} = 2.5$, which is to be expected. However, what was unexpected was the Hard Left Strategy doing better than the Soft Left Strategy. Originally, we hypothesized that the Soft Left Strategy would be faster because you see more faces when you take a soft left as opposed to when you take a hard left. Our results show that this is not the case. A possible explanation for this is it may be more advantageous for the agents to stay relatively close to their initial location than to walk in a new direction and lose their initial area of sight.

The longer multi-step strategies only apply to the icosahedron and dodecahedron under face visibility. These strategies yield a range of results, some with expected meeting times much longer than the unbiased random walk strategies, and some with much smaller expected meeting times. Some strategies have the added property that they guarantee the agents see each other by the end of the sequence, while others actually do worse than the random walk. This variety in performance could potentially be very informative in the formation of an optimal, or nearly-optimal, strategy on the sphere.

We were also able to prove optimality of one of the multi-step strategies on the dodecahedron. To prove this, we consider a lower bound of the expected time, calculated under the assumption that the agents will see each other on the next step if they do not see each other during one iteration of the sequence. We have yet to apply this methodology to multi-step strategies on the icosahe-

dron, but that is something we are looking to do in the future. Moreover, this method could be applied to the cube and the octahedron, providing an alternate proof for the optimality of the Left Strategy on these solids.

In an attempt to further lower the optimal expected meeting time, we consider mixed strategies as a way to insert an asymmetric case. So far, we only have results for mixed strategies composed of two strategies on the dodecahedron. Our work on the dodecahedron suggests that there is potential for a mixed strategy to have a lower expected time than all pure strategies. However, there are many combinations of pure strategies that do not have a beneficial translation as a mixed strategy. In other words, the asymmetric case does not help lower the expected time. In those cases, it is more beneficial for the agents to use a pure strategy as opposed to a mixed strategy. In addition to numerical results, we have many results pertaining to the process of finding the optimal expected time of the set of mixed strategies. Applying these methods to the other solids would be a good step of corroboration for the future.

5.2 Interpretation of Results

Overall, we have a wide variety of results that can be informative in determining the ”best” strategy for different situations. For example, as mentioned before, some strategies guarantee the players see each other in a finite length

of time. These strategies do not always have the lowest expected times. Additionally, for other strategies without this guarantee, the chance of searching infinitely is very small. However, when considering a ”life or death” situation, it may make more sense use a strategy that has a longer expected time, but guarantees rendezvous.

One of our more surprising results is the fact that some strategies have worse expected times than the random walk strategy. One would think that the random walk strategies would give some of the worst possible results since the agents are not trying to strategize. However, as mentioned in the previous section, we found that it is possible to have multi-step strategies that yield significantly worse results than a random walk. This result hints at the notion of a solution for an ”avoidance” problem. For example, consider a search problem where there is a hider and a seeker. These multi-step strategies with longer expected times would be more beneficial for the hider. This variation of the problem could be an interesting topic for further exploration.

One important idea I want to emphasize is that our results are presented in a very general form. For each solid, we consider one unit of length to be one edge length and the agents move at a rate of one edge length per unit of time. But in reality, if all of the solids were embedded in the same size sphere, then the edge length on a cube would be longer than the edge length on a dodecahedron. Therefore, in order to compare the solids, one would need to properly

account for the length differences. In addition, time is an important variable. For example, for mixed strategies, our bounds for the optimal expected time are 2.753707 and 2.8. This results in an interval of length approximately equal to 0.05. At first glance, it may look like at their absolute best, mixed strategies do not perform significantly better than our optimal pure strategy. In one sense, that is a correct conclusion. However, more context is needed in terms of the time unit and what is considered a long length of time. If the time unit is seconds or minutes, then for most situations it is true that the difference is not significant. If the time unit is months or years and the scenario is two people looking for each other, then this difference can be practically significant. Overall, there are many aspects to take into account when addressing a rendezvous search problem.

5.3 Future Research

There are many options for future research on this topic. Topics that we are particularly interested in pursuing are mixed strategies on the other solids, multi-step and mixed strategies under adjacent-vertex visibility (under this visibility, the agents can only see the vertices that are adjacent to the one they are standing on) and no visibility (the agents must meet), and making conjectures for the sphere.

Another possibility for future research is to add more edges and vertices to the faces of the solids to closer approximate the sphere. One obstacle we have found in the first approach is that the solids are no longer vertex-transitive, so there are many cases to be explored and calculations become much more complicated. Therefore, another possibility is to consider other vertex-transitive solids that are larger than the platonic solids. For more information on this work, refer to [9].

Another possibility is to consider strategies with a waiting component. West has considered the case where in the unbiased random walk strategy, the agents also have the option to wait at each step [9]. It would be interesting to see how inserting a waiting component in the multi-step strategies would affect results. In addition, one could adapt the Anderson-Weber strategy for our search and see if it is as effective.

Overall, there are many more strategies to be explored on the platonic solids and other vertex-transitive solids. Comparing these strategies will hopefully point us in the right direction for more complex approximations on the sphere.

Appendix A

Code

A.1 Preface

We wrote functions in Python to perform various calculations, such as finding the probability distribution of the distance between the agents after one iteration of the strategy sequence, calculating the expected time (and bounded expected time) for pure and mixed multi-step strategies, and determining whether a strategy guaranteed the agents seeing each other by the end of the sequence. These programs were only used for calculations on the dodecaedron and icosahedron. In general, the code does not take an incredibly long time to run. However, one possible project for the future is to improve the code and make it more efficient. Another possible project is to write a more general function where the platonic solid is one of the inputs.

A.2 Python Code

Packages that were used:

```
import numpy as np
from numpy.linalg import inv
```

A map of the dodecahedron. We label the vertices 1 through 20. The list, adj_i is a list of the vertices adjacent to vertex i. We then create a list of these adjacency lists.

```
adj_1 = [2,8,3]
adj_2 = [5,1,4]
adj_3 = [1,10,11]
adj_4 = [2,11,7]
adj_5 = [15,2,6]
adj_6 = [17,5,7]
adj_7 = [6,4,19]
adj_8 = [1,15,9]
adj_9 = [8,14,10]
adj_10 = [9,12,3]
adj_11 = [4,3,20]
adj_12 = [10,13,20]
```

```
adj_13 = [14,18,12]

adj_14 = [16,13,9]

adj_15 = [16,8,5]

adj_16 = [15,17,14]

adj_17 = [16,6,18]

adj_18 = [13,17,19]

adj_19 = [18,7,20]

adj_20 = [12,19,11]


connections = [adj_1, adj_2, adj_3, adj_4, adj_5, adj_6,
adj_7, adj_8, adj_9, adj_10, adj_11, adj_12, adj_13,
adj_14, adj_15, adj_16, adj_17, adj_18, adj_19, adj_20]
```

The function minDistance returns the vertex of minimum distance for the given set of unvisited vertices. This is used in the implementation of Dijkstra's algorithm. V is the set of vertices. S is the set of visited vertices. The input, dist is the list of distances from a fixed vertex to all of the other vertices. We update this list as we go through Dijkstra's algorithm.

```
def minDistance(V,S,dist):
    intersect = [] #list of unvisited points
    for i in V:
```

```
            if i not in S:
                intersect.append(i)
        min = 10000
        index = 0
        for n in intersect:
            if dist[n] < min:
                min = dist[n]
                index = n
        return min, index
        #Returns distance of point and index in list
```

The Kalg function is an implementation of Dijkstra's algorithm. We use it to find the shortest distance between two vertices. The inputs place1 and place2 refer to two vertices, and Kalg returns the minimum distance between place1 and place2. This is one portion of the code that takes longer than needed since it calculates the minimum distance for all pairs of vertices involving place1. However, the code still doesn't take too long to run.

```
def Kalg(place1, place2):
    #general list of points
    V = list(range(20))
    #dist is a vector of distances
```

```
dist =list(range(20))
X = 20
W = list(range(20))
W.remove(place1-1)
dist[place1-1] = 0
for v in W:
    dist[v] = 1000
S = [] #list of visited points
while X > 0:
#loop goes while there are still unvisited points
    a, index = mindistance(V,S,dist)
    S.append(index)
    #adds the point from mindistance function
    to list of visited points
    X = X-1
    i_1 = index
    #loop though adjacent points
    for x in connections[i_1]:
        #index for the adjacent points
        i_2 = x-1
        if dist[i_2] > dist[i_1]+1:
```

```
                # reassign to smaller distance
                dist[i_2] = dist[i_1]+1
    return(dist[place2-1])
```

The functions leftDirectionSeq and rightDirectionSeq are used to define a left movement and a right movement on the dodecahedron. The input, place refers to the current location of the agent (location is denoted by the labeling of the vertices that was used when defining the map of the dodecahedron). The input, prev_place refers to the agent's previous location. This is required because left and right movements are relative to where the agent came from. The outputs of the function are the location the agent will be at after the left or right movement, as well as the original location (place).

```
#function to define left movement
def leftDirectionSeq(place, prev_place):
    index = place - 1
    adjacent = connections[index]
    if prev_place == adjacent[0]:
        newplace = adjacent[1]
    elif prev_place == adjacent[1]:
        newplace = adjacent[2]
    else:
```

```
        newplace = adjacent[0]
    return newplace, place


#function to define right movement
def rightDirectionSeq(place, prev_place):
    index = place - 1
    adjacent = connections[index]
    if prev_place == adjacent[0]:
        newplace = adjacent[2]
    elif prev_place == adjacent[1]:
        newplace = adjacent[0]
    else:
        newplace = adjacent[1]
    return newplace, place
```

The following functions are used to calculate the probability distributions of the distance between the agents after executing one iteration of their strategy sequences, as well as the probability of seeing the other agent on a certain step in the sequence.

The function probSeq2 returns a list of probabilities. For an n-step strategy, this list is of length $3n+3$. For each step in the sequence, the probability of being

zero edges apart, the probability of being one edge apart, and the probability of being two edges apart are recorded. In addition, for the last step the probability of being three edges apart, the probability of being four edges apart, and the probability of being five edges apart are also recorded. The input, connections refers to the list of adjacency lists created earlier. The input, k refers to the location of the second agent. (We assume the first agent is at vertex 1. Later on, k is determined by the desired initial distance apart). The inputs seq and seq2 refer to the deterministic sequences of the strategies. These inputs must be lists composed of the elements 'L' and 'R' (e.g. ['L', 'L', 'R']). 'L' denotes left and 'R' denotes right. These lists need not be the same. This function is used later in the function multiStepProb2.

The function multiStepProb32 calculates the probability distribution given that the agents start three edges apart. We consider the case where the agents start three edges apart as separate because there are two probability distributions. The distribution depends on the relative locations of the agents. Thus, this function outputs a list of two lists, each list recording a probability distribution for each case. This function is used later in the function multiStepProb2.

The function multiStepProb2 calculates the probability distribution given that the agents start n edges apart, where $n \in \{3, 4, 5\}$.

```
def probSeq2(connections, k, seq, seq2):
    w = len(seq)
```

```
counts = [0]*((3*(w)+6))
for i in connections[0]:
    for j in connections[k-1]:
        dist = Kalg(i,j)
        if dist==0 or dist==1 or dist==2:
            counts[dist] = counts[dist]+1
        else:
            a = i
            b = j
            z = 1
            y = k
            found = 0
            c = 0
            while found == 0 and c < len(seq):
                if seq[c] == "L":
                    a, z = leftDirectionSeq(a,z)
                else:
                    a, z = rightDirectionSeq(a,z)
                if seq2[c] == "L":
                    b, y = leftDirectionSeq(b,y)
                else:
```

```
                    b, y = rightDirectionSeq(b,y)
                dist = Kalg(a,b)
                if dist==0 or dist==1 or dist==2:
                    found = 1
                c = c+1
            counts[dist+(3*c)] = counts[dist+(3*c)]+1
    probabilities = [x/sum(counts) for x in counts]
    return probabilities



def multiStepProb32(connections, seq, seq2):
    k1 = 6
    k2 = 7
    probabilities1 = probSeq2(connections, k1, seq, seq2)
    probabilities2 = probSeq2(connections, k2, seq, seq2)
    return(probabilities1, probabilities2)
#connections refer to what vertices are adjacent to what
#n refers to starting distance apart
#w refers to how many steps.
For example, if we have one random move
and one left move, then w=2
```

```
#Returns probability distributions for distance
def multiStepProb2(connections, n, seq, seq2):
    if n == 3:
        probabilities
        = multiStepProb32(connections, seq, seq2)
    if n == 4:
        k = 13
        probabilities
        = probSeq2(connections, k, seq, seq2)
    if n == 5:
        k = 18
        probabilities
        = probSeq2(connections, k, seq, seq2)
    return probabilities
```

The function expectedTime returns the expected time. The inputs prob3, prob4, and prob5 refer to the probability distribution lists generated by the function multiStepProb2 for $n = 3$, $n = 4$, and $n = 5$, respectively.

```
def expectedTime(prob3, prob4, prob5):
    prob31 = prob3[0]
```

```
prob32 = prob3[1]
d11=0
d12=0
d2=0
d3 = 0
et = [1,1,1,2,2,2,3,3,3,4,4,4,5,5,5,6,6,6,7,7,7,7,7,7]
x = 24
for i in range(x):
    d11 = d11 + prob31[i]*et[i]
    d12 = d12 + prob32[i]*et[i]
    d2 = d2 + prob4[i]*et[i]
    d3 = d3 + prob5[i]*et[i]
a1 = (1/2)*prob31[-3] + (1/2)*prob32[-3]
a2 = (1/2)*prob31[-2] + (1/2)*prob32[-2]
a3 = (1/2)*prob31[-1] + (1/2)*prob32[-1]
M_11 = 1 - a1
M_12 = -1*a2
M_13 = -1*a3
M_21 = -1*prob4[-3]
M_22 = 1 - prob4[-2]
M_23 = -1*prob4[-1]
```

```
M_31 = -1*prob5[-3]
M_32 = -1*prob5[-2]
M_33 = 1 - prob5[-1]
d1 = (1/2)*d11 + (1/2)*d12
M = np.array([[M_11, M_12, M_13], [M_21, M_22, M_23],
[M_31, M_32, M_33]])
A = inv(np.matrix(M))
v = np.array([d1,d2,d3])
expected_int = np.matmul(A,v)
g=6/10
h=3/10
l=1/10
d = np.matrix([g, h, l])
expected = np.matmul(expected_int, d.transpose())
return expected
```

The function guarantee returns whether there is a guarantee of two agents seeing each other by the end of the first iteration of the strategy sequence. The output states either 'Not Guarantee' or 'Guarantee'. The inputs prob3, prob4, and prob5 refer to the probability distribution lists generated by the function multiStepProb2 for $n = 3$, $n = 4$, and $n = 5$, respectively.

```
def guarantee(prob3, prob4 prob5):
    prob31 = prob3[0]
    prob32 = prob3[1]
    x = 'Not Guarantee'
    if prob31[-1] == 0 and prob31[-2] == 0
    and prob31[-3] == 0:
        if prob32[-1] == 0 and prob32[-2] == 0
        and prob32[-3] == 0:
            if prob4[-1] == 0 and prob4[-2] == 0
            and prob4[-3] == 0:
                if prob5[-1] == 0
                and prob5[-2] == 0
                and prob5[-3] == 0:
                    x = 'Guarantee'
    return(x)
```

The function mixedExpected returns the expected time of a mixed multi-step strategy composed of two pure multi-step strategies. The input, expected_1 refers to the expected time given both agents use strategy 1. The input, expected_12 refers to the expected time given the agents use different strategies. The input, expected_2 refers to the expected time given the agents both use

strategy 2. The function outputs the optimal probability and the final expected time. This function can also be used to calculate the lower bound for the expected time by using lower bounds for the function inputs.

```
def mixedExpected(expected_1, expected_12, expected_2):
    if expected_1 - 2*expected_12 + expected_2 <= 0.0000001:
        if expected_1 < expected_2:
            p=1
        else:
            p=0
    else:
        num = 2*expected_2 - 2*expected_12
        den = 2*expected_1 - 4*expected_12 + 2*expected_2
        p = num/den
        if p < 0 or p > 1:
            if expected_1 < expected_2:
                p=1
            else:
                p=0
    mixexpected = p*p*expected_1 + 2*p*(1-p)*expected_12 +
    (1-p)*(1-p)*expected_2
    return(p, mixexpected)
```

The function boundedExpected returns the (lower) bounded expected time for a pure multi-step strategy. The inputs prob3, prob4, and prob5 refer to the probability distribution lists generated by the function multiStepProb2 for $n = 3$, $n = 4$, and $n = 5$, respectively.

```
def boundedExpected(prob3, prob4, prob5):
    prob31 = prob3[0]
    prob32 = prob3[1]
    d11=0
    d12=0
    d2=0
    d3=0
    et = [1,1,1,2,2,2,3,3,3,4,4,4,5,5,5,6,6,6,7,7,7,8,8,8]
    #Assume that if they don't see each other,
    they see each other on 8th step
    x = 24
    for i in range(x):
        d11 = d11 + prob31[i]*et[i]
        d12 = d12 + prob32[i]*et[i]
        d2 = d2 + prob4[i]*et[i]
        d3 = d3 + prob5[i]*et[i]
    d1 = (1/2)*d11 + (1/2)*d12
```

```
    v = np.array([d1,d2,d3])
    g = 6/10
    h = 3/10
    l = 1/10
    d = np.matrix([g, h, l])
    expected = np.matmul(v ,d.transpose())
    return expected
```

A map of the icosahedron. We label the vertices 1 through 12. The list, adj_i is a list of the vertices adjacent to vertex i. We then create a list of these adjacency lists.

```
# Map of the icosahedron
adj_1 = [6,2,3,4,5]
adj_2 = [7,8,3,1,6]
adj_3 = [2,8,9,4,1]
adj_4 = [1,3,9,10,5]
adj_5 = [6,1,4,10,11]
adj_6 = [5,11,7,2,1]
adj_7 = [12,8,2,6,11]
adj_8 = [12,9,3,2,7]
adj_9 = [3,8,12,10,4]
```

```
adj_10 = [4,9,12,11,5]

adj_11 = [10,12,7,6,5]

adj_12 = [9,8,7,11,10]


connections = [adj_1, adj_2, adj_3, adj_4, adj_5, adj_6,

adj_7, adj_8, adj_9, adj_10, adj_11, adj_12]
```

The functions hardLeftDirectionSeq, hardRightDirectionSeq, softLeftDirectionSeq, and softRightDirectionSeq are used to define a hard left movement, hard right movement, soft left movement, soft right movement on the icosahedron, respectively. The input, place refers to the current location of the agent (location is denoted by the labeling of the vertices that was used when defining the map of the icosahedron). The input, prev_place refers to the agent's previous location. This is required because the movements are relative to where the agent came from. The outputs of the function are the location the agent will be at after the movement, as well as the original location (place).

```
#function to define hard left movement on icosahedron

def hardLeftDirectionSeq(place, prev_place):

    index = place - 1

    adjacent = connections[index]

    if prev_place == adjacent[0]:
```

```
        newplace = adjacent[1]
    elif prev_place == adjacent[1]:
        newplace = adjacent[2]
    elif prev_place == adjacent[2]:
        newplace = adjacent[3]
    elif prev_place == adjacent[3]:
        newplace = adjacent[4]
    else:
        newplace = adjacent[0]
    return newplace, place

#function to define hard right movement on icosahedron
def hardRightDirectionSeq(place, prev_place):
    index = place - 1
    adjacent = connections[index]
    if prev_place == adjacent[0]:
        newplace = adjacent[4]
    elif prev_place == adjacent[1]:
        newplace = adjacent[0]
    elif prev_place == adjacent[2]:
        newplace = adjacent[1]
```

```
    elif prev_place == adjacent[3]:
        newplace = adjacent[2]
    else:
        newplace = adjacent[3]
    return newplace, place


#function to define soft left movement on icosahedron
def softLeftDirectionSeq(place, prev_place):
    index = place - 1
    adjacent = connections[index]
    if prev_place == adjacent[0]:
        newplace = adjacent[2]
    elif prev_place == adjacent[1]:
        newplace = adjacent[3]
    elif prev_place == adjacent[2]:
        newplace = adjacent[4]
    elif prev_place == adjacent[3]:
        newplace = adjacent[0]
    else:
        newplace = adjacent[1]
    return newplace, place
```

```
#function to define soft right movement on icosahedron
def softRightDirectionSeq(place, prev_place):
    index = place - 1
    adjacent = connections[index]
    if prev_place == adjacent[0]:
        newplace = adjacent[3]
    elif prev_place == adjacent[1]:
        newplace = adjacent[4]
    elif prev_place == adjacent[2]:
        newplace = adjacent[0]
    elif prev_place == adjacent[3]:
        newplace = adjacent[1]
    else:
        newplace = adjacent[2]
    return newplace, place
```

Bibliography

[1] L. D. Stone, *Theory of Optimal Search*. Academic Press, New York, 1975.

[2] S. Gal, *Search Games*. Academic Press, New York, 1980.

[3] T. Shelling, *The Strategy of Conflict*. Harvard University Press, Cambridge, 1960.

[4] S. Alpern and S. Gal, *The Theory of Search Games and Rendezvous*. New York: Springer, 2011.

[5] E. J. Anderson and R. R. Weber, "The rendezvous problem on discrete locations," *Journal of Applied Probability*, vol. 27, pp. 839–851, 1990.

[6] R. Weber, "Optimal symmetric rendezvous search on three locations," *Mathematics of Operations Research*, vol. 37, pp. 111–122, 2012.

[7] S. Alpern, "The rendezvous search problem," *SIAM Journal on Control and Optimization*, vol. 33, pp. 673–683, 1995.

BIBLIOGRAPHY

[8] N. S. E. Kranakis, D. Krizanc and C. Sawchuk, “Mobile agent rendezvous in a ring,” *Proc. 23rd Internation Conference on Distributed Computing Systems (ICDCS 2003)*, pp. 592–599, 2003.

[9] E. West, “Rendezvous search on the edges of vertex-transitive solids,” Master’s thesis, Johns Hopkins University, Baltimore, MD, 2018.

[10] J. W. E. West, X. Xie, “Rendezvous search on the edges of platonic solids: Optimal waiting probabilities and multi-step strategies,” *Congressus Numerantium*, 2018, to appear.

Vita

Xiao (Annie) Xie is from Guilford, Connecticut. She is a student in the combined Bachelor's Master's program in Applied Mathematics and Statistics at Johns Hopkins University. Her academic interests include probability, statistics, optimization, and STEM education. Her other interests include travelling, painting, and reading. In the future, Annie hopes to pursue a Ph.D. degree in either Applied Mathematics or Statistics.